Solving simple PDEs using the finite element method

FEM for simple PDEs: elliptic and parabolic linear PDEs

PDEs

Second-order PDEs commonly arise in physical models. There are three archetypal second-order PDEs

1. Elliptic PDEs, for example, Poisson's equation $\nabla^2 u + f = 0$
2. Parabolic PDEs, for example, the heat equation $u_t = \nabla^2 u + f$
3. Hyperbolic PDEs, for example, the wave equation $u_{tt} = \nabla^2 u$

PDEs

Second-order PDEs commonly arise in physical models. There are three archetypal second-order PDEs

1. **Elliptic PDEs, for example, Poisson's equation** $\nabla^2 u + f = 0$
2. **Parabolic PDEs, for example, the heat equation** $u_t = \nabla^2 u + f$
3. Hyperbolic PDEs, for example, the wave equation $u_{tt} = \nabla^2 u$

Defining PDEs in an object oriented manner

First, an abstract class defining a general linear elliptic PDE $\nabla \cdot D\boldsymbol{\nabla} u = f$, where D is a matrix-valued function of position (the diffusion tensor):

`AbstractLinearEllipticPde`:
- *Abs. method:* `GetDiffusionTensor(x)`
- *Abs. method:* `GetForceTerm(x)`

`MyEllipticPde`: *inherits from* `AbstractLinearEllipticPde`
- *Implemented method:* `GetDiffusionTensor(x)`
- *Implemented method:* `GetForceTerm(x)`

For example $\nabla^2 u = 0$

`LaplacesEquation`: *inherits from* `AbstractLinearEllipticPde`
- *Implemented method:* `GetDiffusionTensor(x)`
  - ▷ *return identity matrix*
- *Implemented method:* `GetForceTerm(x)`
  - ▷ *return zero*

Defining PDEs in an object oriented manner

First, an abstract class defining a general linear elliptic PDE $\nabla \cdot D\boldsymbol{\nabla} u = f$, where D is a matrix-valued function of position (the diffusion tensor):

`AbstractLinearEllipticPde`:
  Abs. method: `GetDiffusionTensor(x)`
  Abs. method: `GetForceTerm(x)`

`MyEllipticPde`: *inherits from* `AbstractLinearEllipticPde`
  Implemented method: `GetDiffusionTensor(x)`
  Implemented method: `GetForceTerm(x)`

For example $\nabla^2 u = 0$

`LaplacesEquation`: *inherits from* `AbstractLinearEllipticPde`
  Implemented method: `GetDiffusionTensor(x)`
    ▷ *return identity matrix*
  Implemented method: `GetForceTerm(x)`
    ▷ *return zero*

Defining PDEs in an object oriented manner

First, an abstract class defining a general linear elliptic PDE $\nabla \cdot D\boldsymbol{\nabla} u = f$, where D is a matrix-valued function of position (the diffusion tensor):

`AbstractLinearEllipticPde`:
- *Abs. method:* `GetDiffusionTensor(x)`
- *Abs. method:* `GetForceTerm(x)`

`MyEllipticPde`: *inherits from* `AbstractLinearEllipticPde`
- *Implemented method:* `GetDiffusionTensor(x)`
- *Implemented method:* `GetForceTerm(x)`

For example $\nabla^2 u = 0$

`LaplacesEquation`: *inherits from* `AbstractLinearEllipticPde`
- *Implemented method:* `GetDiffusionTensor(x)`
  - ▷ *return identity matrix*
- *Implemented method:* `GetForceTerm(x)`
  - ▷ *return zero*

Defining PDEs in an object oriented manner

Next, an abstract class defining a general linear parabolic PDE

$$\alpha u_t = \nabla \cdot D \boldsymbol{\nabla} u + f$$

where α, D and f are functions of space and time.

`AbstractLinearParabolicPde`:

Abs. method: `GetDuDtCoefficientTerm(t,x)`
Abs. method: `GetDiffusionTensor(t,x)`
Abs. method: `GetForceTerm(t,x)`

For example $u_t = \nabla^2 u$

`HeatEquation`: *inherits from* `AbstractLinearParabolicPde`

Implemented method: `GetDuDtCoefficientTerm(t,x)`
▷ *return 1*
Implemented method: `GetDiffusionTensor(x)`
▷ *return identity matrix*
Implemented method: `GetForceTerm(x)`
▷ *return zero*

Defining PDEs in an object oriented manner

Next, an abstract class defining a general linear parabolic PDE

$$\alpha u_t = \nabla \cdot D\boldsymbol{\nabla} u + f$$

where α, D and f are functions of space and time.

`AbstractLinearParabolicPde`:

Abs. method: `GetDuDtCoefficientTerm(t,x)`
Abs. method: `GetDiffusionTensor(t,x)`
Abs. method: `GetForceTerm(t,x)`

For example $u_t = \nabla^2 u$

`HeatEquation`: *inherits from* `AbstractLinearParabolicPde`

Implemented method: `GetDuDtCoefficientTerm(t,x)`
▷ *return 1*
Implemented method: `GetDiffusionTensor(x)`
▷ *return identity matrix*
Implemented method: `GetForceTerm(x)`
▷ *return zero*

FEM for simple PDEs: introduction to FEM

The finite element method

Stages

1. Convert equation from **strong form** to **weak form**
2. Convert infinite-dimensional problem into a finite dimensional one
3. Set up the finite element linear system to be solved

Weak form of Poisson's equation

Consider Poisson's equation:

$$\nabla^2 u + f = 0$$

subject to boundary conditions

$$\begin{aligned} u &= 0 \quad \text{on } \Gamma_1 \\ \boldsymbol{\nabla} u \cdot \mathbf{n} &= g \quad \text{on } \Gamma_2 \end{aligned}$$

Weak form

Multiply by a test function v satisfying $v = 0$ on Γ_1, and integrate:

$$\begin{aligned} v\left(\nabla^2 u\right) &= -fv \\ \int_\Omega v\left(\nabla^2 u\right) \mathrm{d}V &= -\int_\Omega fv \, \mathrm{d}V \\ \int_{\partial\Omega} v\left(\boldsymbol{\nabla} u \cdot \mathbf{n}\right) \mathrm{d}S - \int_\Omega \boldsymbol{\nabla} u \cdot \boldsymbol{\nabla} v \, \mathrm{d}V &= -\int_\Omega fv \, \mathrm{d}V \\ \int_\Omega \boldsymbol{\nabla} u \cdot \boldsymbol{\nabla} v \, \mathrm{d}V &= \int_\Omega fv \, \mathrm{d}V + \int_{\Gamma_2} gv \, \mathrm{d}S \end{aligned}$$

Weak form of Poisson's equation

Consider Poisson's equation:

$$\nabla^2 u + f = 0$$

subject to boundary conditions

$$\begin{aligned} u &= 0 \quad \text{on } \Gamma_1 \\ \boldsymbol{\nabla} u \cdot \mathbf{n} &= g \quad \text{on } \Gamma_2 \end{aligned}$$

Weak form

Multiply by a test function v satisfying $v = 0$ on Γ_1, and integrate:

$$\begin{aligned} v\left(\nabla^2 u\right) &= -fv \\ \int_\Omega v\left(\nabla^2 u\right) \mathrm{d}V &= -\int_\Omega fv \,\mathrm{d}V \\ \int_{\partial\Omega} v\,(\boldsymbol{\nabla} u \cdot \mathbf{n})\,\mathrm{d}S - \int_\Omega \boldsymbol{\nabla} u \cdot \boldsymbol{\nabla} v \,\mathrm{d}V &= -\int_\Omega fv \,\mathrm{d}V \\ \int_\Omega \boldsymbol{\nabla} u \cdot \boldsymbol{\nabla} v \,\mathrm{d}V &= \int_\Omega fv \,\mathrm{d}V + \int_{\Gamma_2} gv \,\mathrm{d}S \end{aligned}$$

Weak form of Poisson's equation

Let $\mathcal{V}$ be the space of all differentiable functions on Ω (more precisely, $\mathcal{V}$ is the Sobolev space $H^1(\Omega)$). Let

$$\mathcal{V}_0 = \{v \in \mathcal{V} \,:\, v = 0 \text{ on } \Gamma_1\}$$

Weak form

Find $u \in \mathcal{V}_0$ satisfying

$$\int_\Omega \nabla u \cdot \nabla v \,\mathrm{d}V = \int_\Omega fv \,\mathrm{d}V + \int_{\Gamma_2} gv \,\mathrm{d}S \qquad \forall v \in \mathcal{V}_0$$

Example

Solve $\frac{\mathrm{d}^2 u}{\mathrm{d}x^2} = 1$, $u(0) = u(1) = 0$ vs Find differentiable u satisfying $u(0) = u(1) = 0$ and: $\int_0^1 \frac{\mathrm{d}u}{\mathrm{d}x}\frac{\mathrm{d}v}{\mathrm{d}x}\,\mathrm{d}x = -\int_0^1 v\,\mathrm{d}x$ for all v s.t. $v(0) = v(1) = 0$

Weak form of Poisson's equation

Let $\mathcal{V}$ be the space of all differentiable functions on Ω (more precisely, $\mathcal{V}$ is the Sobolev space $H^1(\Omega)$). Let

$$\mathcal{V}_0 = \{v \in \mathcal{V} \,:\, v = 0 \text{ on } \Gamma_1\}$$

Weak form

Find $u \in \mathcal{V}_0$ satisfying

$$\int_\Omega \nabla u \cdot \nabla v \, \mathrm{d}V = \int_\Omega f v \, \mathrm{d}V + \int_{\Gamma_2} g v \, \mathrm{d}S \qquad \forall v \in \mathcal{V}_0$$

Example

Solve $\frac{\mathrm{d}^2 u}{\mathrm{d}x^2} = 1$, $u(0) = u(1) = 0$ vs Find differentiable u satisfying $u(0) = u(1) = 0$ and: $\int_0^1 \frac{\mathrm{d}u}{\mathrm{d}x}\frac{\mathrm{d}v}{\mathrm{d}x}\,\mathrm{d}x = -\int_0^1 v\,\mathrm{d}x$ for all v s.t. $v(0) = v(1) = 0$

Weak form of Poisson's equation

Let $\mathcal{V}$ be the space of all differentiable functions on Ω (more precisely, $\mathcal{V}$ is the Sobolev space $H^1(\Omega)$). Let

$$\mathcal{V}_0 = \{v \in \mathcal{V} \,:\, v = 0 \text{ on } \Gamma_1\}$$

Weak form

Find $u \in \mathcal{V}_0$ satisfying

$$\int_\Omega \nabla u \cdot \nabla v \,\mathrm{d}V = \int_\Omega fv \,\mathrm{d}V + \int_{\Gamma_2} gv \,\mathrm{d}S \qquad \forall v \in \mathcal{V}_0$$

Example

Solve $\frac{\mathrm{d}^2 u}{\mathrm{d}x^2} = 1$, $u(0) = u(1) = 0$

vs

Find differentiable u satisfying $u(0) = u(1) = 0$ and: $\int_0^1 \frac{\mathrm{d}u}{\mathrm{d}x}\frac{\mathrm{d}v}{\mathrm{d}x}\,\mathrm{d}x = -\int_0^1 v\,\mathrm{d}x$ for all v s.t. $v(0) = v(1) = 0$

FEM discretisation

Find $u \in \mathcal{V}_0$ satisfying

$$\int_\Omega \nabla u \cdot \nabla v \,\mathrm{d}V = \int_\Omega f v \,\mathrm{d}V + \int_{\Gamma_2} g v \,\mathrm{d}S \qquad \text{for all } v \in \mathcal{V}_0$$

Take

$$\mathcal{V}_0^h = \mathrm{span}\{\phi_1, \phi_2\}$$

(where ϕ_1, ϕ_2 satisfy the Dirichlet boundary conditions), so

$$u_h = \alpha\phi_1 + \beta\phi_2$$

Linear system:

$$\begin{bmatrix} \int_\Omega \nabla\phi_1 \cdot \nabla\phi_1 \,\mathrm{d}V & \int_\Omega \nabla\phi_1 \cdot \nabla\phi_2 \,\mathrm{d}V \\ \int_\Omega \nabla\phi_2 \cdot \nabla\phi_1 \,\mathrm{d}V & \int_\Omega \nabla\phi_2 \cdot \nabla\phi_2 \,\mathrm{d}V \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \int_\Omega f\phi_1 \,\mathrm{d}V + \int_{\Gamma_2} g\phi_1 \,\mathrm{d}S \\ \int_\Omega f\phi_2 \,\mathrm{d}V + \int_{\Gamma_2} g\phi_2 \,\mathrm{d}S \end{bmatrix}$$

FEM discretisation

Find $u_h \in \mathcal{V}_0^h$ satisfying

$$\int_\Omega \nabla u_h \cdot \nabla v \,\mathrm{d}V = \int_\Omega f v \,\mathrm{d}V + \int_{\Gamma_2} g v \,\mathrm{d}S \qquad \text{for all } v \in \mathcal{V}_0^h$$

Take

$$\mathcal{V}_0^h = \mathrm{span}\{\phi_1, \phi_2\}$$

(where ϕ_1, ϕ_2 satisfy the Dirichlet boundary conditions), so

$$u_h = \alpha\phi_1 + \beta\phi_2$$

Linear system:

$$\begin{bmatrix} \int_\Omega \nabla\phi_1 \cdot \nabla\phi_1 \,\mathrm{d}V & \int_\Omega \nabla\phi_1 \cdot \nabla\phi_2 \,\mathrm{d}V \\ \int_\Omega \nabla\phi_2 \cdot \nabla\phi_1 \,\mathrm{d}V & \int_\Omega \nabla\phi_2 \cdot \nabla\phi_2 \,\mathrm{d}V \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \int_\Omega f\phi_1 \,\mathrm{d}V + \int_{\Gamma_2} g\phi_1 \,\mathrm{d}S \\ \int_\Omega f\phi_2 \,\mathrm{d}V + \int_{\Gamma_2} g\phi_2 \,\mathrm{d}S \end{bmatrix}$$

FEM discretisation

Find $u_h \in \mathcal{V}_0^h$ satisfying

$$\int_\Omega \nabla u_h \cdot \nabla v \, \mathrm{d}V = \int_\Omega fv \, \mathrm{d}V + \int_{\Gamma_2} gv \, \mathrm{d}S \qquad \text{for all } v \in \mathcal{V}_0^h$$

Take

$$\mathcal{V}_0^h = \mathrm{span}\{\phi_1, \phi_2\}$$

(where ϕ_1, ϕ_2 satisfy the Dirichlet boundary conditions), so

$$u_h = \alpha\phi_1 + \beta\phi_2$$

Linear system:

$$\begin{bmatrix} \int_\Omega \nabla\phi_1 \cdot \nabla\phi_1 \, \mathrm{d}V & \int_\Omega \nabla\phi_1 \cdot \nabla\phi_2 \, \mathrm{d}V \\ \int_\Omega \nabla\phi_2 \cdot \nabla\phi_1 \, \mathrm{d}V & \int_\Omega \nabla\phi_2 \cdot \nabla\phi_2 \, \mathrm{d}V \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \int_\Omega f\phi_1 \, \mathrm{d}V + \int_{\Gamma_2} g\phi_1 \, \mathrm{d}S \\ \int_\Omega f\phi_2 \, \mathrm{d}V + \int_{\Gamma_2} g\phi_2 \, \mathrm{d}S \end{bmatrix}$$

FEM discretisation

Find $u_h \in \mathcal{V}_0^h$ satisfying

$$\int_\Omega \nabla u_h \cdot \nabla \phi_j \,\mathrm{d}V = \int_\Omega f \phi_j \,\mathrm{d}V + \int_{\Gamma_2} g \phi_j \,\mathrm{d}S \qquad \text{for } j = 1, 2$$

Take

$$\mathcal{V}_0^h = \mathrm{span}\{\phi_1, \phi_2\}$$

(where ϕ_1, ϕ_2 satisfy the Dirichlet boundary conditions), so

$$u_h = \alpha \phi_1 + \beta \phi_2$$

Linear system:

$$\begin{bmatrix} \int_\Omega \nabla \phi_1 \cdot \nabla \phi_1 \,\mathrm{d}V & \int_\Omega \nabla \phi_1 \cdot \nabla \phi_2 \,\mathrm{d}V \\ \int_\Omega \nabla \phi_2 \cdot \nabla \phi_1 \,\mathrm{d}V & \int_\Omega \nabla \phi_2 \cdot \nabla \phi_2 \,\mathrm{d}V \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \int_\Omega f \phi_1 \,\mathrm{d}V + \int_{\Gamma_2} g \phi_1 \,\mathrm{d}S \\ \int_\Omega f \phi_2 \,\mathrm{d}V + \int_{\Gamma_2} g \phi_2 \,\mathrm{d}S \end{bmatrix}$$

FEM discretisation

Find $u_h \in \mathcal{V}_0^h$ satisfying

$$\int_\Omega \nabla u_h \cdot \nabla \phi_j \,\mathrm{d}V = \int_\Omega f \phi_j \,\mathrm{d}V + \int_{\Gamma_2} g \phi_j \,\mathrm{d}S \qquad \text{for } j = 1, 2$$

Take

$$\mathcal{V}_0^h = \mathrm{span}\{\phi_1, \phi_2\}$$

(where ϕ_1, ϕ_2 satisfy the Dirichlet boundary conditions), so

$$u_h = \alpha\phi_1 + \beta\phi_2$$

Linear system:

$$\begin{bmatrix} \int_\Omega \nabla \phi_1 \cdot \nabla \phi_1 \,\mathrm{d}V & \int_\Omega \nabla \phi_1 \cdot \nabla \phi_2 \,\mathrm{d}V \\ \int_\Omega \nabla \phi_2 \cdot \nabla \phi_1 \,\mathrm{d}V & \int_\Omega \nabla \phi_2 \cdot \nabla \phi_2 \,\mathrm{d}V \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \int_\Omega f \phi_1 \,\mathrm{d}V + \int_{\Gamma_2} g \phi_1 \,\mathrm{d}S \\ \int_\Omega f \phi_2 \,\mathrm{d}V + \int_{\Gamma_2} g \phi_2 \,\mathrm{d}S \end{bmatrix}$$

FEM discretisations

Take

$$V_h = \mathrm{span}\{\phi_1, \phi_2, \ldots, \phi_N\}$$

(satisfying $\phi_j = 0$ on Γ_1) so

$$u_h = \alpha_1 \phi_1 + \ldots + \alpha_N \phi_N$$

Let the stiffness matrix and RHS vector be given by

$$\begin{aligned} K_{jk} &= \int_\Omega \boldsymbol{\nabla} \phi_j \cdot \boldsymbol{\nabla} \phi_k \, \mathrm{d}V \\ b_j &= \int_\Omega f \phi_j \, \mathrm{d}V + \int_{\Gamma_2} g \phi_j \, \mathrm{d}S \end{aligned}$$

and solve

$$K \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_N \end{bmatrix} = \mathbf{b}$$

Basis functions

FEM discretisations

Let

$$
\begin{aligned}
K_{jk} &= \int_\Omega \boldsymbol{\nabla}\phi_j \cdot \boldsymbol{\nabla}\phi_k \,\mathrm{d}V \qquad \text{stiffness matrix} \\
M_{jk} &= \int_\Omega \phi_j \phi_k \,\mathrm{d}V \qquad \text{mass matrix} \\
b_j &= \int_\Omega f \phi_j \,\mathrm{d}V + \int_{\Gamma_2} g\phi_j \,\mathrm{d}S
\end{aligned}
$$

FEM discretisations

Laplace's equation: $\nabla^2 u + f = 0 \quad \rightarrow \quad K\mathbf{U} = \mathbf{b}$

Heat equation:

$$\frac{\partial u}{\partial t} = \nabla^2 u + f \quad \rightarrow \quad M\frac{\mathrm{d}\mathbf{U}}{\mathrm{d}t} + K\mathbf{U} = \mathbf{b}$$

Time-discretised heat equation:

$$\frac{u^{n+1} - u^n}{\Delta t} = \nabla^2 u^{n+1} + f^{n+1} \quad \rightarrow \quad M\mathbf{U}^{n+1} + \Delta t\, K\mathbf{U}^{n+1} = M\mathbf{U}^n + \Delta t\,\mathbf{b}^{n+1}$$

Anisotropic diffusion

Suppose we have an anisotropic diffusion tensor D (symmetric, positive definite), for example, in Poisson's equation:

$$\nabla \cdot (D \boldsymbol{\nabla} u) + f = 0$$

subject to boundary conditions

$$\begin{aligned} u &= 0 && \text{on } \Gamma_1 \\ (D \boldsymbol{\nabla} u) \cdot \mathbf{n} &= g && \text{on } \Gamma_2 \end{aligned}$$

The weak form is: find $u \in \mathcal{V}_0$ satisfying

$$\int_\Omega (D \boldsymbol{\nabla} u) \cdot \boldsymbol{\nabla} v \, \mathrm{d}V = \int_\Omega f v \, \mathrm{d}V + \int_{\Gamma_2} g v \, \mathrm{d}S \qquad \forall v \in \mathcal{V}_0$$

and the only change in the FEM discretisation is that the stiffness matrix becomes

$$K_{jk} = \int_\Omega \boldsymbol{\nabla} \phi_j \cdot (D \boldsymbol{\nabla} \phi_k) \, \mathrm{d}V$$

Anisotropic diffusion

Suppose we have an anisotropic diffusion tensor D (symmetric, positive definite), for example, in Poisson's equation:

$$\nabla \cdot (D \boldsymbol{\nabla} u) + f = 0$$

subject to boundary conditions

$$\begin{aligned} u &= 0 \qquad \text{on } \Gamma_1 \\ (D \boldsymbol{\nabla} u) \cdot \mathbf{n} &= g \qquad \text{on } \Gamma_2 \end{aligned}$$

The weak form is: find $u \in \mathcal{V}_0$ satisfying

$$\int_\Omega (D \boldsymbol{\nabla} u) \cdot \boldsymbol{\nabla} v \, \mathrm{d}V = \int_\Omega f v \, \mathrm{d}V + \int_{\Gamma_2} g v \, \mathrm{d}S \qquad \forall v \in \mathcal{V}_0$$

and the only change in the FEM discretisation is that the stiffness matrix becomes

$$K_{jk} = \int_\Omega \boldsymbol{\nabla} \phi_j \cdot (D \boldsymbol{\nabla} \phi_k) \, \mathrm{d}V$$

Implementing Dirichlet boundary conditions

In practice, rather using the basis functions in $\mathcal{V}_0^h$ (i.e. bases satisfying $\phi_i = 0$ on Γ_1), we use $\mathcal{V}^h$, i.e. all the basis functions corresponding to all nodes in the mesh.

We then impose (any) Dirichlet boundary conditions by altering the appropriate rows of the linear system, for example, for $K\mathbf{U} = b$, if we want to impose $U_1 = c$

$$\begin{bmatrix} K_{11} & K_{12} & \dots & K_{1N} \\ K_{21} & K_{22} & \dots & K_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ K_{N1} & K_{N2} & \dots & K_{NN} \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \\ \vdots \\ U_N \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_N \end{bmatrix}$$

Implementing Dirichlet boundary conditions

In practice, rather using the basis functions in $\mathcal{V}_0^h$ (i.e. bases satisfying $\phi_i = 0$ on Γ_1), we use $\mathcal{V}^h$, i.e. all the basis functions corresponding to all nodes in the mesh.

We then impose (any) Dirichlet boundary conditions by altering the appropriate rows of the linear system, for example, for $K\mathbf{U} = b$, if we want to impose $U_1 = c$

$$\begin{bmatrix} K_{11} & K_{12} & \dots & K_{1N} \\ K_{21} & K_{22} & \dots & K_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ K_{N1} & K_{N2} & \dots & K_{NN} \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \\ \vdots \\ U_N \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_N \end{bmatrix}$$

Implementing Dirichlet boundary conditions

In practice, rather using the basis functions in $\mathcal{V}_0^h$ (i.e. bases satisfying $\phi_i = 0$ on Γ_1), we use $\mathcal{V}^h$, i.e. all the basis functions corresponding to all nodes in the mesh.

We then impose (any) Dirichlet boundary conditions by altering the appropriate rows of the linear system, for example, for $K\mathbf{U} = b$, if we want to impose $U_1 = c$

$$\begin{bmatrix} 1 & 0 & \dots & 0 \\ K_{21} & K_{22} & \dots & K_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ K_{N1} & K_{N2} & \dots & K_{NN} \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \\ \vdots \\ U_N \end{bmatrix} = \begin{bmatrix} c \\ b_2 \\ \vdots \\ b_N \end{bmatrix}$$

FEM stages

Solve:

$$\nabla \cdot (D \boldsymbol{\nabla} u) + f = 0$$

subject to boundary conditions

$$\begin{aligned} u &= u^* && \text{on } \Gamma_1 \\ (D \boldsymbol{\nabla} u) \cdot \mathbf{n} &= g && \text{on } \Gamma_2 \end{aligned}$$

1. Set up the computational mesh and choose basis functions
2. Compute the matrix K and vector $\mathbf{b}$:

$$\begin{aligned} K_{jk} &= \int_\Omega \boldsymbol{\nabla}\phi_j \cdot (D \boldsymbol{\nabla}\phi_k) \, \mathrm{d}V \\ b_j &= \int_\Omega f \phi_j \, \mathrm{d}V + \int_{\Gamma_2} g \phi_j \, \mathrm{d}S \end{aligned}$$

3. Alter linear system $K\mathbf{U} = \mathbf{b}$ to impose Dirichlet BCs
4. Solve linear system

FEM for simple PDEs: FEM details

Computing a finite element matrix/vector by *assembly*

Consider computing the mass matrix $M_{jk} = \int_\Omega \phi_j \phi_k \, \mathrm{d}V$, an N by N matrix say, and let's suppose (for clarity only) that we are in 2D. Also, assume we are using linear basis functions.

We **do not** write out the full basis functions explicitly in computing this integral. Instead: firstly, we break the integral down into an integral over elements:

$$M_{jk} = \sum_{\mathcal{K}} \int_{\mathcal{K}} \phi_j \phi_k \, \mathrm{d}V$$

Consider $\int_{\mathcal{K}} \phi_j \phi_k \, \mathrm{d}V$. **Key point**: The only basis functions with are non-zero in the triangle are the 3 basis functions corresponding to the 3 nodes of the element.

Therefore: compute the **elemental contribution to the mass matrix**, a 3 by 3 matrix of the form $\int_{\mathcal{K}} \phi_j \phi_k \, \mathrm{d}V$ for *3 choices of j and k only*.

Then **add** elemental contribution to full N by N mass matrix.

Computing a finite element matrix/vector by *assembly*

Consider computing the mass matrix $M_{jk} = \int_\Omega \phi_j \phi_k \, \mathrm{d}V$, an N by N matrix say, and let's suppose (for clarity only) that we are in 2D. Also, assume we are using linear basis functions.

We **do not** write out the full basis functions explicitly in computing this integral. Instead: firstly, we break the integral down into an integral over elements:

$$M_{jk} = \sum_{\mathcal{K}} \int_{\mathcal{K}} \phi_j \phi_k \, \mathrm{d}V$$

Consider $\int_{\mathcal{K}} \phi_j \phi_k \, \mathrm{d}V$. **Key point**: The only basis functions with are non-zero in the triangle are the 3 basis functions corresponding to the 3 nodes of the element.

Therefore: compute the **elemental contribution to the mass matrix**, a 3 by 3 matrix of the form $\int_{\mathcal{K}} \phi_j \phi_k \, \mathrm{d}V$ for *3 choices of j and k only*.

Then **add** elemental contribution to full N by N mass matrix.

Computing a finite element matrix/vector by *assembly*

Consider computing the mass matrix $M_{jk} = \int_\Omega \phi_j \phi_k \, \mathrm{d}V$, an N by N matrix say, and let's suppose (for clarity only) that we are in 2D. Also, assume we are using linear basis functions.

We **do not** write out the full basis functions explicitly in computing this integral. Instead: firstly, we break the integral down into an integral over elements:

$$M_{jk} = \sum_{\mathcal{K}} \int_{\mathcal{K}} \phi_j \phi_k \, \mathrm{d}V$$

Consider $\int_{\mathcal{K}} \phi_j \phi_k \, \mathrm{d}V$. **Key point**: The only basis functions with are non-zero in the triangle are the 3 basis functions corresponding to the 3 nodes of the element.

Therefore: compute the **elemental contribution to the mass matrix**, a 3 by 3 matrix of the form $\int_{\mathcal{K}} \phi_j \phi_k \, \mathrm{d}V$ for *3 choices of j and k only*.

Then **add** elemental contribution to full N by N mass matrix.

Computing a finite element matrix/vector by *assembly*

Consider computing the mass matrix $M_{jk} = \int_\Omega \phi_j \phi_k \, \mathrm{d}V$, an N by N matrix say, and let's suppose (for clarity only) that we are in 2D. Also, assume we are using linear basis functions.

We **do not** write out the full basis functions explicitly in computing this integral. Instead: firstly, we break the integral down into an integral over elements:

$$M_{jk} = \sum_{\mathcal{K}} \int_{\mathcal{K}} \phi_j \phi_k \, \mathrm{d}V$$

Consider $\int_{\mathcal{K}} \phi_j \phi_k \, \mathrm{d}V$. **Key point**: The only basis functions with are non-zero in the triangle are the 3 basis functions corresponding to the 3 nodes of the element.

Therefore: compute the **elemental contribution to the mass matrix**, a 3 by 3 matrix of the form $\int_{\mathcal{K}} \phi_j \phi_k \, \mathrm{d}V$ for *3 choices of j and k only*.

Then **add** elemental contribution to full N by N mass matrix.

Computing a finite element matrix/vector by *assembly*

Consider computing the mass matrix $M_{jk} = \int_\Omega \phi_j \phi_k \,\mathrm{d}V$, an N by N matrix say, and let's suppose (for clarity only) that we are in 2D. Also, assume we are using linear basis functions.

We **do not** write out the full basis functions explicitly in computing this integral. Instead: firstly, we break the integral down into an integral over elements:

$$M_{jk} = \sum_{\mathcal{K}} \int_{\mathcal{K}} \phi_j \phi_k \,\mathrm{d}V$$

Consider $\int_{\mathcal{K}} \phi_j \phi_k \,\mathrm{d}V$. **Key point**: The only basis functions with are non-zero in the triangle are the 3 basis functions corresponding to the 3 nodes of the element.

Therefore: compute the **elemental contribution to the mass matrix**, a 3 by 3 matrix of the form $\int_{\mathcal{K}} \phi_j \phi_k \,\mathrm{d}V$ for *3 choices of j and k only*.

Then **add** elemental contribution to full N by N mass matrix.

Computing an elemental contribution

We have reduced the problem to computing small matrices/vectors, for example the 3 by 3 matrix

$$\int_{\mathcal{K}} \phi_j \phi_k \, \mathrm{d}V$$

where ϕ_j, ϕ_k are the 3 basis functions corresponding to the 3 nodes of the mesh.

Next, map to the **reference triangle** (also known as the canonical triangle), $\mathcal{K}_{\mathrm{ref}}$, the triangle with nodes $(0,0)$, $(0,1)$, $(1,0)$.

The basis functions on the reference triangle are easy to write down

$$\begin{aligned} N_1(\xi,\eta) &= 1-\xi-\eta \\ N_2(\xi,\eta) &= \xi \\ N_3(\xi,\eta) &= \eta \end{aligned}$$

Computing an elemental contribution

We now need to be able to compute

$$\int_{\mathcal{K}} \phi_j \phi_k \,\mathrm{d}x\mathrm{d}y = \int_{\mathcal{K}_{\mathrm{ref}}} N_j N_k \det J \,\mathrm{d}\xi\mathrm{d}\eta$$

where J is the Jacobian of the mapping from the true element to the canonical element.

J is also required if $\nabla\phi_i$ is needed (for example, in computing the stiffness matrix), since $\nabla\phi_i = J\nabla_\xi N_i$.

Consider the mapping from an element with nodes $\mathbf{x}_1$, $\mathbf{x}_2$, $\mathbf{x}_3$, to the canonical element. The inverse mapping can in fact be easily written down using the basis functions.

$$\mathbf{x}(\xi,\eta) = \sum_{j=1}^{3} \mathbf{x}_j N_j(\xi,\eta)$$

from which it is easy to show that J is the following function of nodal positions

$$J = \mathrm{inv}\begin{bmatrix} x_2 - x_1 & x_3 - x_1 \\ y_2 - y_1 & y_3 - y_1 \end{bmatrix}$$

Computing an elemental contribution

We now need to be able to compute

$$\int_{\mathcal{K}} \phi_j \phi_k \,\mathrm{d}x\mathrm{d}y = \int_{\mathcal{K}_{\mathrm{ref}}} N_j N_k \det J \,\mathrm{d}\xi \mathrm{d}\eta$$

where J is the Jacobian of the mapping from the true element to the canonical element.

J is also required if $\nabla \phi_i$ is needed (for example, in computing the stiffness matrix), since $\nabla \phi_i = J \nabla_\xi N_i$.

Consider the mapping from an element with nodes $\mathbf{x}_1$, $\mathbf{x}_2$, $\mathbf{x}_3$, to the canonical element. The inverse mapping can in fact be easily written down using the basis functions.

$$\mathbf{x}(\xi, \eta) = \sum_{j=1}^{3} \mathbf{x}_j N_j(\xi, \eta)$$

from which it is easy to show that J is the following function of nodal positions

$$J = \mathrm{inv} \begin{bmatrix} x_2 - x_1 & x_3 - x_1 \\ y_2 - y_1 & y_3 - y_1 \end{bmatrix}$$

Computing an elemental contribution

We now need to be able to compute

$$\int_{\mathcal{K}} \phi_j \phi_k \, \mathrm{d}x\mathrm{d}y = \int_{\mathcal{K}_{\mathrm{ref}}} N_j N_k \det J \, \mathrm{d}\xi \mathrm{d}\eta$$

where J is the Jacobian of the mapping from the true element to the canonical element.

J is also required if $\nabla \phi_i$ is needed (for example, in computing the stiffness matrix), since $\nabla \phi_i = J \nabla_{\xi} N_i$.

Consider the mapping from an element with nodes $\mathbf{x}_1$, $\mathbf{x}_2$, $\mathbf{x}_3$, to the canonical element. The inverse mapping can in fact be easily written down using the basis functions.

$$\mathbf{x}(\xi, \eta) = \sum_{j=1}^{3} \mathbf{x}_j N_j(\xi, \eta)$$

from which it is easy to show that J is the following function of nodal positions

$$J = \mathrm{inv} \begin{bmatrix} x_2 - x_1 & x_3 - x_1 \\ y_2 - y_1 & y_3 - y_1 \end{bmatrix}$$

Computing an elemental contribution - the general case

Suppose we want to compute

$$\int_{\mathcal{K}} \mathcal{F}(x, y, u, \phi_1, \phi_2, \phi_3, \boldsymbol{\nabla}\phi_1, \boldsymbol{\nabla}\phi_2, \boldsymbol{\nabla}\phi_3)\, \mathrm{d}x\mathrm{d}y$$

We map to the reference element:

$$\int_{\mathcal{K}_{\mathrm{ref}}} \mathcal{F}(x, y, u, \phi_1, \phi_2, \phi_3, \boldsymbol{\nabla}\phi_1, \boldsymbol{\nabla}\phi_2, \boldsymbol{\nabla}\phi_3)\, \det J \; \mathrm{d}\xi\mathrm{d}\eta$$

and then use **numerical quadrature**, which means f just has to be evaluated at the quadrature points.

FEM stages - full algorithm

Solve:

$$\nabla \cdot (D \boldsymbol{\nabla} u) + f = 0$$

subject to boundary conditions

$$\begin{aligned} u &= u^* && \text{on } \Gamma_1 \\ (D \boldsymbol{\nabla} u) \cdot \mathbf{n} &= g && \text{on } \Gamma_2 \end{aligned}$$

1. Set up the computational mesh and choose basis functions
2. Compute the matrix K and vector $\mathbf{b}$:

$$\begin{aligned} K_{jk} &= \int_\Omega \boldsymbol{\nabla} \phi_j \cdot (D \boldsymbol{\nabla} \phi_k) \, \mathrm{d}V \\ b_j &= \int_\Omega f \phi_j \, \mathrm{d}V + \int_{\Gamma_2} g \phi_j \, \mathrm{d}S \end{aligned}$$

3. Alter linear system $K\mathbf{U} = \mathbf{b}$ to impose Dirichlet BCs
4. Solve linear system

FEM stages - full algorithm

Write

$$b_j = \int_\Omega f \phi_j \,\mathrm{d}V + \int_{\Gamma_2} g \phi_j \,\mathrm{d}S$$

as $\mathbf{b} = \mathbf{b}^{\text{vol}} + \mathbf{b}^{\text{surf}}$

1. Set up the computational mesh and choose basis functions
2. Compute the matrix K and vector $\mathbf{b}$:
    1. Loop over elements, for each compute the elemental contributions K_{elem} and $\mathbf{b}^{\text{vol}}_{\mathrm{elem}}$ (3 by 3 matrix and 3-vector)
        - For this, need to compute Jacobian J for this element, and loop over quadrature points.
    2. Add K_{elem} and $\mathbf{b}^{\text{vol}}_{\mathrm{elem}}$ to K and $\mathbf{b}^{\text{vol}}$ appropriately
    3. Loop over surface-elements on Γ_2, for each compute the elemental contribution $\mathbf{b}^{\text{surf}}_{\mathrm{elem}}$ (a 2-vector).
    4. Add $\mathbf{b}^{\text{surf}}_{\mathrm{elem}}$ to $\mathbf{b}^{\text{surf}}$ appropriately
3. Alter linear system $K\mathbf{U} = \mathbf{b}$ to impose Dirichlet BCs
4. Solve linear system

FEM stages - full algorithm

Write

$$b_j = \int_\Omega f \phi_j \, \mathrm{d}V + \int_{\Gamma_2} g \phi_j \, \mathrm{d}S$$

as $\mathbf{b} = \mathbf{b}^{\mathrm{vol}} + \mathbf{b}^{\mathrm{surf}}$

1. Set up the computational mesh and choose basis functions
2. Compute the matrix K and vector $\mathbf{b}$:
    1. Loop over elements, for each compute the elemental contributions K_{elem} and $\mathbf{b}^{\mathrm{vol}}_{\mathrm{elem}}$ (3 by 3 matrix and 3-vector)
        - For this, need to compute Jacobian J for this element, and loop over quadrature points
    2. Add K_{elem} and $\mathbf{b}^{\mathrm{vol}}_{\mathrm{elem}}$ to K and $\mathbf{b}^{\mathrm{vol}}$ appropriately
    3. Loop over surface-elements on Γ_2, for each compute the elemental contribution $\mathbf{b}^{\mathrm{surf}}_{\mathrm{elem}}$ (a 2-vector).
    4. Add $\mathbf{b}^{\mathrm{surf}}_{\mathrm{elem}}$ to $\mathbf{b}^{\mathrm{surf}}$ appropriately
3. Alter linear system $K\mathbf{U} = \mathbf{b}$ to impose Dirichlet BCs
4. Solve linear system

FEM stages - full algorithm

Write

$$b_j = \int_\Omega f \phi_j \, \mathrm{d}V + \int_{\Gamma_2} g \phi_j \, \mathrm{d}S$$

as $\mathbf{b} = \mathbf{b}^{\text{vol}} + \mathbf{b}^{\text{surf}}$

1. Set up the computational mesh and choose basis functions
2. Compute the matrix K and vector $\mathbf{b}$:
    1. Loop over elements, for each compute the elemental contributions K_{elem} and $\mathbf{b}^{\text{vol}}_{\mathrm{elem}}$ (3 by 3 matrix and 3-vector)
        - For this, need to compute Jacobian J for this element, and loop over quadrature points
    2. Add K_{elem} and $\mathbf{b}^{\text{vol}}_{\mathrm{elem}}$ to K and $\mathbf{b}^{\text{vol}}$ appropriately
    3. Loop over surface-elements on Γ_2, for each compute the elemental contribution $\mathbf{b}^{\text{surf}}_{\mathrm{elem}}$ (a 2-vector).
        - Similar to integrals over elements, again use quadrature
    4. Add $\mathbf{b}^{\text{surf}}_{\mathrm{elem}}$ to $\mathbf{b}^{\text{surf}}$ appropriately
3. Alter linear system $K\mathbf{U} = \mathbf{b}$ to impose Dirichlet BCs
4. Solve linear system

FEM stages - full algorithm

Write

$$b_j = \int_\Omega f\phi_j \,\mathrm{d}V + \int_{\Gamma_2} g\phi_j \,\mathrm{d}S$$

as $\mathbf{b} = \mathbf{b}^{\text{vol}} + \mathbf{b}^{\text{surf}}$

1. Set up the computational mesh and choose basis functions
2. Compute the matrix K and vector $\mathbf{b}$:
    1. Loop over elements, for each compute the elemental contributions K_{elem} and $\mathbf{b}^{\text{vol}}_{\text{elem}}$ (3 by 3 matrix and 3-vector)
        - For this, need to compute Jacobian J for this element, and loop over quadrature points
    2. Add K_{elem} and $\mathbf{b}^{\text{vol}}_{\text{elem}}$ to K and $\mathbf{b}^{\text{vol}}$ appropriately
    3. Loop over surface-elements on Γ_2, for each compute the elemental contribution $\mathbf{b}^{\text{surf}}_{\text{elem}}$ (a 2-vector).
        - Similar to integrals over elements, again use quadrature
    4. Add $\mathbf{b}^{\text{surf}}_{\text{elem}}$ to $\mathbf{b}^{\text{surf}}$ appropriately
3. Alter linear system $K\mathbf{U} = \mathbf{b}$ to impose Dirichlet BCs
4. Solve linear system

FEM stages - full algorithm

Write

$$b_j = \int_{\Omega} f \phi_j \, \mathrm{d}V + \int_{\Gamma_2} g \phi_j \, \mathrm{d}S$$

as $\mathbf{b} = \mathbf{b}^{\text{vol}} + \mathbf{b}^{\text{surf}}$

1. Set up the computational mesh and choose basis functions
2. Compute the matrix K and vector $\mathbf{b}$:
   1. Loop over elements, for each compute the elemental contributions K_{elem} and $\mathbf{b}^{\text{vol}}_{\text{elem}}$ (3 by 3 matrix and 3-vector)
      - For this, need to compute Jacobian J for this element, and loop over quadrature points
   2. Add K_{elem} and $\mathbf{b}^{\text{vol}}_{\text{elem}}$ to K and $\mathbf{b}^{\text{vol}}$ appropriately
   3. Loop over surface-elements on Γ_2, for each compute the elemental contribution $\mathbf{b}^{\text{surf}}_{\text{elem}}$ (a 2-vector).
      - Similar to integrals over elements, again use quadrature
   4. Add $\mathbf{b}^{\text{surf}}_{\text{elem}}$ to $\mathbf{b}^{\text{surf}}$ appropriately
3. Alter linear system $K\mathbf{U} = \mathbf{b}$ to impose Dirichlet BCs
4. Solve linear system

FEM stages - full algorithm

Write

$$b_j = \int_{\Omega} f \phi_j \, \mathrm{d}V + \int_{\Gamma_2} g \phi_j \, \mathrm{d}S$$

as $\mathbf{b} = \mathbf{b}^{\mathrm{vol}} + \mathbf{b}^{\mathrm{surf}}$

1. Set up the computational mesh and choose basis functions
2. Compute the matrix K and vector $\mathbf{b}$:
    1. Loop over elements, for each compute the elemental contributions K_{elem} and $\mathbf{b}^{\mathrm{vol}}_{\mathrm{elem}}$ (3 by 3 matrix and 3-vector)
        - For this, need to compute Jacobian J for this element, and loop over quadrature points
    2. Add K_{elem} and $\mathbf{b}^{\mathrm{vol}}_{\mathrm{elem}}$ to K and $\mathbf{b}^{\mathrm{vol}}$ appropriately
    3. Loop over surface-elements on Γ_2, for each compute the elemental contribution $\mathbf{b}^{\mathrm{surf}}_{\mathrm{elem}}$ (a 2-vector).
        - Similar to integrals over elements, again use quadrature
    4. Add $\mathbf{b}^{\mathrm{surf}}_{\mathrm{elem}}$ to $\mathbf{b}^{\mathrm{surf}}$ appropriately
3. Alter linear system $K\mathbf{U} = \mathbf{b}$ to impose Dirichlet BCs
4. Solve linear system

FEM stages - full algorithm

Write

$$b_j = \int_\Omega f\phi_j \,\mathrm{d}V + \int_{\Gamma_2} g\phi_j \,\mathrm{d}S$$

as $\mathbf{b} = \mathbf{b}^{\mathsf{vol}} + \mathbf{b}^{\mathsf{surf}}$

1. Set up the computational mesh and choose basis functions
2. Compute the matrix K and vector $\mathbf{b}$:
    1. Loop over elements, for each compute the elemental contributions K_{elem} and $\mathbf{b}^{\mathsf{vol}}_{\mathrm{elem}}$ (3 by 3 matrix and 3-vector)
        - For this, need to compute Jacobian J for this element, and loop over quadrature points
    2. Add K_{elem} and $\mathbf{b}^{\mathsf{vol}}_{\mathrm{elem}}$ to K and $\mathbf{b}^{\mathsf{vol}}$ appropriately
    3. Loop over surface-elements on Γ_2, for each compute the elemental contribution $\mathbf{b}^{\mathsf{surf}}_{\mathrm{elem}}$ (a 2-vector).
        - Similar to integrals over elements, again use quadrature
    4. Add $\mathbf{b}^{\mathsf{surf}}_{\mathrm{elem}}$ to $\mathbf{b}^{\mathsf{surf}}$ appropriately
3. Alter linear system $K\mathbf{U} = \mathbf{b}$ to impose Dirichlet BCs
4. Solve linear system